Uniwersytet Warszawski
Wydział Fizyki

Marcin Polkowski
Nr albumu: 251328

Simulation of wave fronts and amplitudes of P and S-waves in spherically symmetric Earth model using ray approach

Praca licencjacka
na kierunku Fizyka

Praca wykonana pod kierunkiem
prof. dr hab. Marka Grada
Zakład Fizyki Litosfery
Instytut Geofizyki
Wydział Fizyki UW

Warszawa, Luty 2012

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

data *Podpis kierującego pracą*

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

data *Podpis autora pracy*

Abstract

Paper presents ray approach to seismic wave tracing in spherically symmetric Earth model. Simulation algorithm is discussed with its implementation in C++ and other modern programming languages. Paper shows possible extensions of implementation to support tracking wave amplitude. Different scientifically and educationally important result are presented and discussed.

Słowa kluczowe

Fale sejmiczne, symulacja propagcji fal sejsmicznych, model jednowymiarowy

Dziedzina pracy (kody wg programu Socrates-Erasmus)

(13.2) Fizyka

Tytuł pracy w języku angielskim

Simulation of wave fronts and amplitudes of P and S-waves in spherically symmetric Earth model using ray approach

Contents

List of Figures

Chapter 1

Introduction

The interior of the Earth has very complicated structure, and its details are still poorly discovered. There are numerous methods of surveying the inside of the Earth, but most of this methods can reach only to the shallowest layers. Among others the following methods are available [1]:

- Boreholes - most precise method that allow studying material with sub millimeter accuracy. Downside of boreholes is small depth range - the deepest borehole has 12,345 km in depth. Regional range of boreholes is also limited. Drilling is extremely expensive so this method is no longer widely used.
- Gravimetry - measurement of the strength of a gravitational field allow to map differences in Earth gravity field that correspond to different rocks density. Interpretation of gravitational data is complicated and ambiguous work. As in case of boreholes depth range is also limited.
- Magnetotellurics - measurement of electric and magnetic anomalies. This method allows to survey subsurface structure up to 30 km deep.

There are other surveying method reaching at most meters or tens of meters in depth: ground penetrating radar, electrical resistivity tomography.

Earth is an extremely large object measuring 6371 km in radius. That means that methods mentioned above let scientist to study less than 1.5% of Earth's volume. Analyzing seismic waves is the only method of surveying whole Earth.

Seismic surveying allows scientists to create models of the Earth's interior [2]. These models are always a simplification of real structure. There are several types of models: 1D (one dimensional, examples: [6, 3, 4]), 2D (example: [5]), 3D (example: [7]) or even more detailed anisotropic 3D [9].

For the whole Earth only two of those are useful: 1D, where parameters depend only on depth (or radius) and 3D, where parameters depend on geographic location and depth. 2D models can be used for local profiles where parameters depend on position on the profile and depth. Anisotropic 3D models require large amount of experimental data, so they can only by created for local areas.

In this paper 1D model of spherically symmetric Earth will be discussed. There are several parameters that can by analyzed and put into Earths model such as: P-wave speed, S-wave speed, pressure, density, Lamé parameters etc. These parameters are related between each other. Because in seismic methods wave traveltime is the measured parameter and distance is known, P and S wave speeds are fundamental parameters for all seismic models.

As mentioned above a seismologist's job is to improve model accuracy and quality for better understanding Earth interior. This process requires, among others, simulating wave propagation through the model itself. This is one of multiple reasons for developing different algorithms of wave propagation simulation. This paper discusses algorithm working in 1D Earth model. Possible application of algorithm also will be discussed.

Chapter 2

1D Earth model

Despite the fact that 1D models are extreme simplification of Earth structure they are useful and valuable. Deep structure of Earth is spherically symmetric with good approximation while shallow layers differentiate with location (for example continent vs. ocean). 1D model are constructed as mean value approximation. 1D model describes Earth as a perfect sphere.

Among many different 1D Earth models two are most commonly used: Preliminary Reference Earth Model (PREM) and *iasp'91*. Construction of PREM was considered a milestone in seismology. Because of that PREM will be used for wave simulating further in this paper.

P and S-wave speed described in PREM are shown on figure 2.1.

Earth's interior is composed of layers within which parameters change smoothly. However on layers boundaries changes might by rapid and step-like. There are several important boundaries with elastic parameters discontinuity that cause reflection and wave conversions described in section 3.3. Among others there are significant boundaries: between inner and outer core, which is liquid (S-wave does not propagate in liquid), and between outer core and mantle. Influence of mentioned discontinuities to wave propagation can be observed on figure 6.1 on page 11.

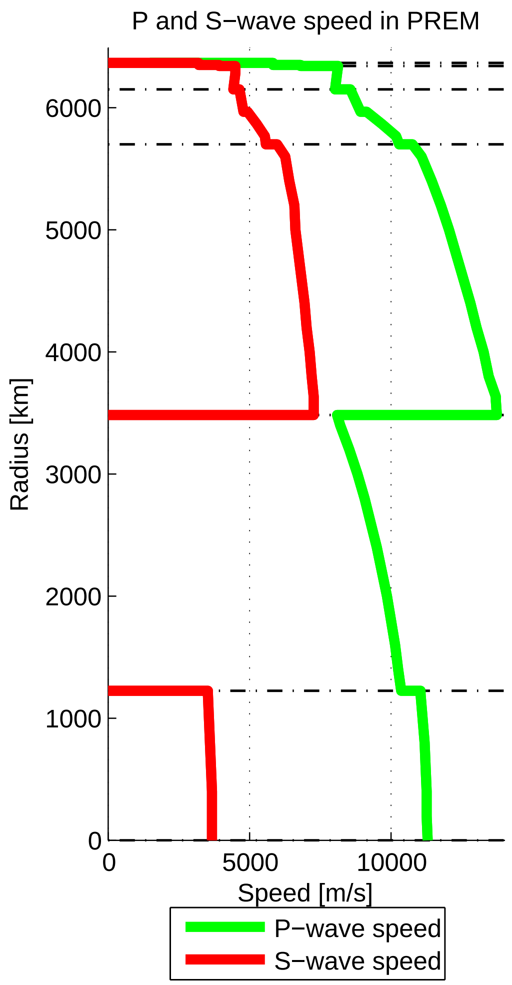

Figure 2.1
P and S-wave speed in Preliminary Reference Earth Model. Dashed lines indicate layers boundaries.

Chapter 3

Simulation method

Methods of simulating wave propagation can be divided into two groups: those based on solving differential equations and others.

Because in general a model is a set of discrete values analytical solutions for differential equations are not possible. Numerical methods have to be introduced: finite element method or finite difference method. Please refer to other sources for detailed explanation of this methods.

Other method (used in this paper) is ray approach [10] - method of tracking path of single ray among model, where it can be refracted, reflected or converted. Ray is always perpendicular to wave front and has mathematical zero-width. It might be compared to laser beam.

There are several significant differences between this methods:

- Ray approach does not simulate wave effects such as diffraction
- Ray approach requires additional steps to simulate amplitude due to energy dissipation
- Ray approach allow to track any wave phase across model (for example tracking PKPPcP phase)

In general finite differences and finite elements are more general methods, but ray approach brings better understanding of processes that are involved during wave propagation: refraction, reflection, conversion [2, 10].

3.1 Refraction

Seismic wave traveling through media obeys Fermat's principle - path taken between two points in medium is the path of minimal traveltime (in general extreme traveltime). This principal is widely used in optics as Snell's Law:

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{v_1}{v_2}$$

Where θ_1 is angle of incidence, θ_2 refracted angle, and v_1, v_2 wave speeds in first and second medium. Notice that refraction occurs everywhere where wave speed changes, not only at layers boundaries.

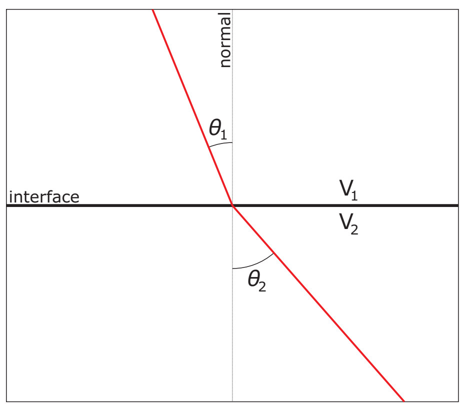

Figure 3.1
Snell's law

3.2 Reflection

Always some of wave energy is being reflected from the interface. If critical angle of incidence is exceeded all energy is being reflected from the interface. Such a situation occurs when $\sin\theta_1 > \frac{v_1}{v_2}$. Reflection is observed at layers boundaries only.

3.3 Conversion

Refraction and reflection are sufficient when there is only one type of wave in the medium. While simulating seismic waves propagation two wave types has to be processed: P-wave and S-wave.

In continuous medium those two types of waves can propagate independently, while on interface with contrast of elastic parameters they can transform P-wave $\rightleftarrows$ S-wave. Figure 3.2 shows possible conversions for solid - solid interface, while figure 3.3 shows conversions for solid - liquid interface.

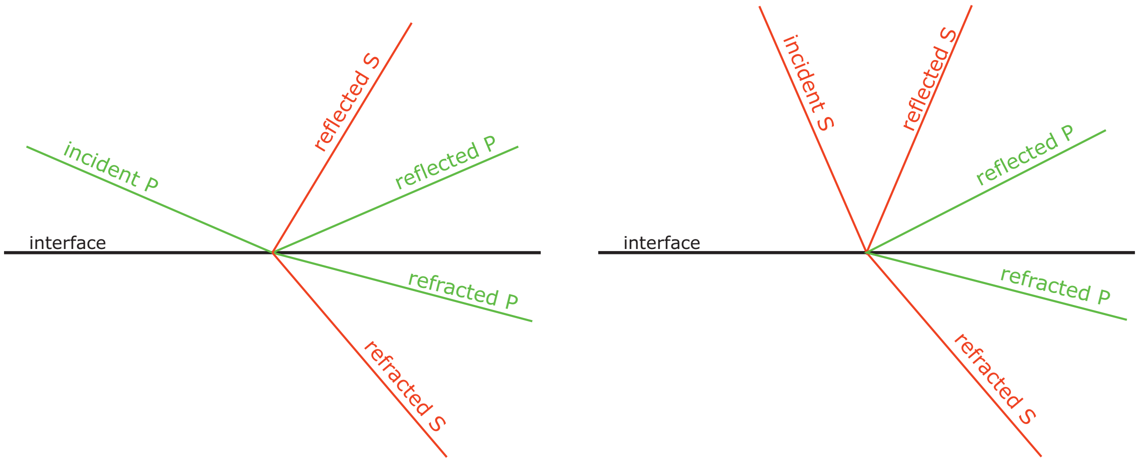

Figure 3.2
At interface with contrast of elastic parameters wave conversion occurs. Figure shows possible conversions for solid-solid interface. One incident wave is converted to up to four new waves: reflected, refracted, converted reflected and converted refracted. Depending on situation some of this waves may not be created (for example when critical angle is exceeded) or may have negligibly small energy.

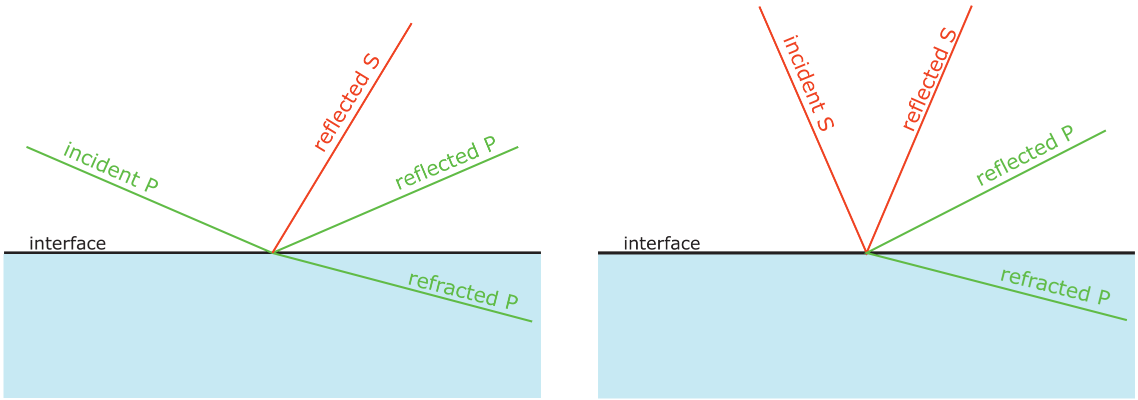

Figure 3.3
Possible conversions for solid-liquid interface. S-waves cannot propagate in liquid medium.

Chapter 4

Algorithm

Tracing of seismic ray in 1D model can be achieved by an iterative numeric algorithm. For ray to by calculated several parameters needs to be given:

- 1D speed model - algorithm needs to calculate speed value at current depth
- Type of wave - P and S waves are calculated separately
- Coordinates of starting point
- Direction of ray propagation (at starting point)
- Calculating step in seconds

Algorithm is divided into two significant parts: ray calculating and control. The role of control algorithm is to invoke ray calculation with proper arguments and collect its results.

Implementation in C++ is shown in listing A.1 on page 17.

Single ray calculation is terminated when one or more of following conditions are satisfied:

- Reached surface
- Reached layer boundary

After termination results (coordinates, direction, time) are returned to main control algorithm, which decides how to process results. At this moment all wave transforms need to be calculated. Depending on situation up to four new rays might be created at this point. Parameters of new rays are placed in queue to by calculated.

The control algorithm need to have some limits. Without those program execution would be infinite (at the end of one ray at least one ray is created). There are number of possible limits depending on expected results:

- Traveltime limit (if one wants to study only rays propagating no longer then given time)
- Conversions limit (if one is interested only in refracted rays, or reflected from certain layer only)
- Amplitude (energy) limit (if one wants to study only rays with amplitudes above given value)
- Other (for example combination of mentioned above)

Chapter 5

Implementation

Several implementations of described algorithm were prepared in different modern programming languages for performance comparison:

MATLAB - the best environment for developing code. It's flexibility allows to make quick analysis of changes in algorithm or its implementation. Various data visualization tools help with understanding the results. MATLAB code was definitely the slowest one - up to 15 times slower than C++. Finally MATLAB was used for preparing plots of computed data.

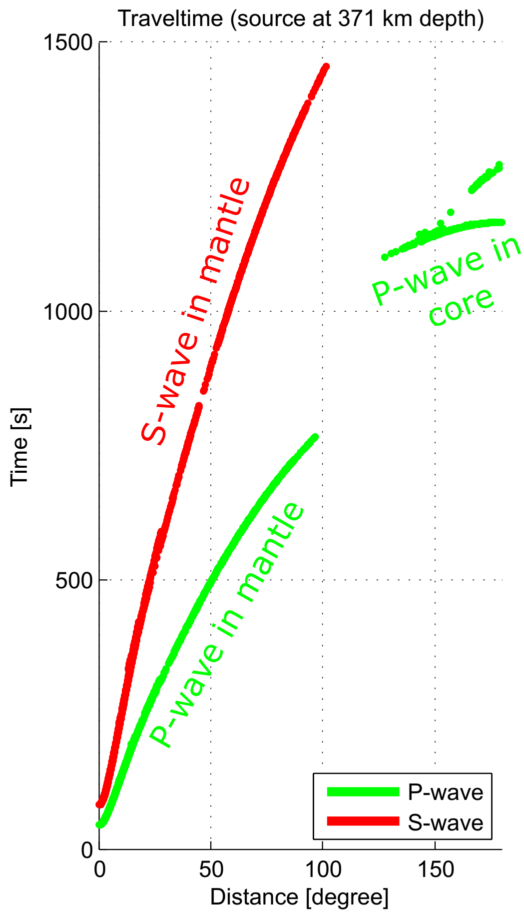

Figure 5.1
Traveltime (source at 371 km depth).

FORTRAN - traditional programming language for solving numeric problems. This implementation was done to verify opinion, that FORTRAN is faster than any other high level programming language. Finally proved to be as fast as C++, but not noticeably faster.

C++ - popular and efficient programming language, current standard for different applications. Final implementation was done in C++ and tested on various platforms including PC/Windows, PC/Linux and even ARM/linux.

C for CUDA - this implementation was done for experimental reasons only. CUDA - Compute Unified Device Architecture is computing engine in graphic processing unit by Nvidia. Graphics processors are parallel units (up to 2048 simultaneous threads per processor) . Ray tracing algorithm can by parallelized (each thread computes one ray, multiple rays are computed at the same time). Implementation of ray tracing algorithm took over two weeks of work, but results was outstanding: over 700 times faster than computer CPU. This computations where done in Interdisciplinary Center for Mathematical and Computational Modeling (ICM) at University of Warsaw within grant number G44-26.

Implementations of the algorithm may vary due to different output and expected result. Different result examples are shown in next chapter.

One of the simplest implementations of the algorithm results in computing traveltime data for P and S-wave in function of epicentral distance (0-180 deg). Source code of this implementation in

C++ is shown on listing in appendix A. Selectable number of rays are calculated (ex. 1024 rays). Half of them are P-wave, half S-wave. Ray starting directions are linearly spaced between 0 and 360 degree. Each ray is propagated until it reaches surface. At this moment time and angular distance are saved to file. Results are shown on figure 5.1.

5.1 Possible expansions

As mentioned earlier implementation of the algorithm can be modified to suit any needs connected with ray tracing of seismic wave. Two very important modifications are reflection / transmission coefficients and amplitude simulation.

Amplitude (or energy) decreases because of two different physical phenomenons: geometrical spreading of wave front and damping. Both can be included in every iteration of ray tracing. If energy or amplitude is calculated it can be used as limit for ending computations. Calculating effect of damping requires additional information in model - Q factor.

Reflection and transmission coefficients can be calculated for each of transformed waves. Formulas are complicated and different for all situations (solid - solid, liquid - solid, solid - liquid and even liquid - liquid). There are several different implementations and simplification of this formulas [2]. In general coefficients are parameters of speeds, densities on both sides of interface and angle of incident. Figure 5.2 shows example relation of coefficients values to incident angle.

Examples of both possible expansions will be shown I chapter 6.

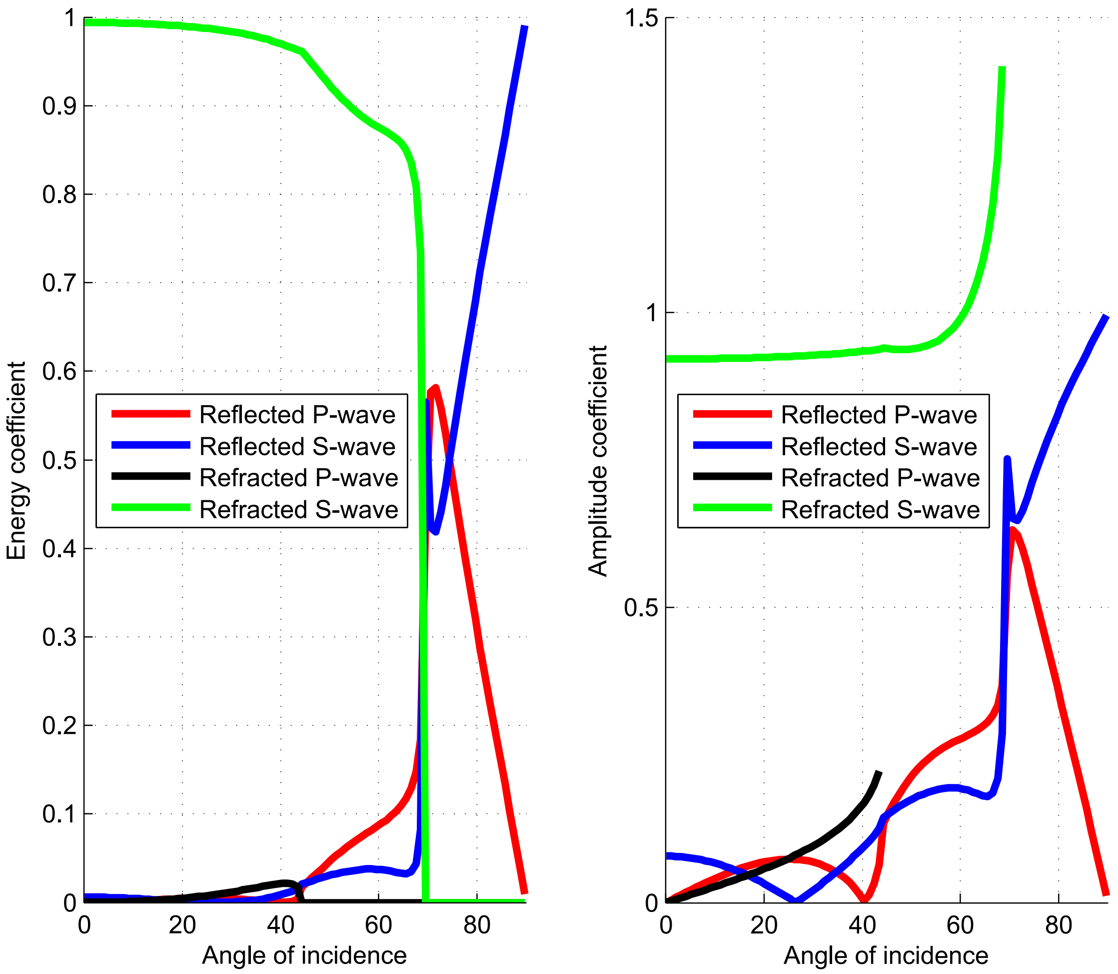

Figure 5.2
Amplitude and energy coefficients calculated for incident S-wave at boundary between Transition Zone and Lower Mantle (so called "670 km" boundary) [8]. Note that some energy (and amplitude) is always reflected, while refracted waves exist only with incident angles under critical. Energy coefficients always sum up to 1 (no energy is lost for conversion itself), however amplitude coefficients may exceed 1, as in given example.

5.2 Performance and accuracy

Discussed iterative algorithm has accuracy that depend on two factors: type of data used in implementation (single or double precision) and length of time step.

While first is machine dependent, second factor can be changed by user. At every program iteration one new ray coordinate is calculated. Length of step depends on wave speed and mentioned time step (distance equals speed times time).

Figure 5.3 shows results for three different time steps: 1 second (square), 0.5 second (triangle), 0.05 second (dot). All rays calculated in the same medium where speeds increases with depth. The shortest time step, the better ray path accuracy.

Ray calculation consumes 99% of total execution time. Every ray step takes the same amount of processor time. As result of that overall execution time depend mainly on chosen time step (the dependence is linear).

Other important factor that influence performance is size of output data. A shown in example in section 6.1 size of data saved during execution my by much bigger then operating memory size. In that case saving to hard drive is involved, which is not efficient.

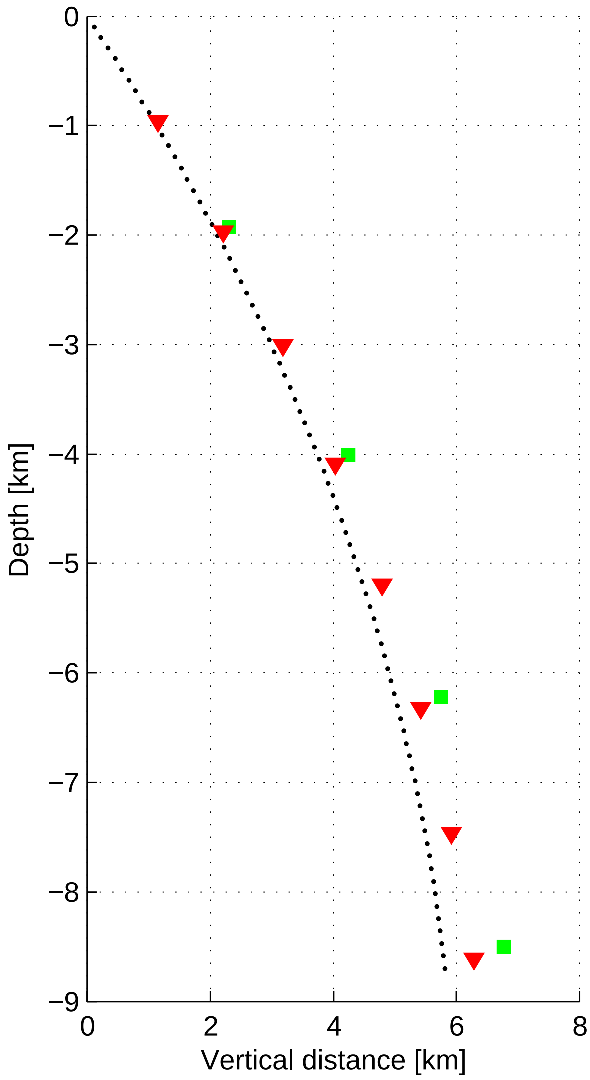

Figure 5.3
Accuracy dependence on time step example. See text for details.

As mentioned before construction of algorithm allows use of parallel computing techniques. While Nvidia CUDA is ultimate parallel solution, implementation and compatibility is hard. Fortunately other methods of parallelization are available: for example OpenMP. This methods are much simpler to implement (and does not require special hardware like CUDA), but performance gain factor is only 2 or 3 for standard workstation [1].

[1]Intel Core I7 (2.93 GHz) processor with 8GB of operating memory and Windows 2008 Server operating system.

Chapter 6

Results

There are multiple output possibilities, depending on need. Useful information can be extracted from simulation for various further analysis. In this chapter three examples are presented. Corresponding source codes are available for request - please contact the author.

6.1 Example 1

In this example full simulation of wave propagation was computed. All converted rays were processed with corresponding reflection / transmission coefficients.

This computation took over 24 hours on standard workstation computer. Algorithm was modified to record information about every wave front position and amplitude every second for 2500 seconds from source time.

Record file was over 110 GB in size. Additional program was prepared to split record file to smaller files, each containing information about rays positions and amplitudes in one second. 2500 files were generated.

Data prepared this way made it possible to prepare animation of wave propagation inside Earth. Animation was used multiple times for educational purposes during lectures (for example "Physics of the Earth's interior" - lecture by prof. Grad for second and third year students).

Figure 6.1 shows four stages of wave propagation: after 250, 500, 750, 1000 seconds from source time.

6.2 Example 2

Code described in section 6.1 was modified to record only data about rays reaching surface instead of all rays data. This information allows creation of traveltime chart (see figure 5.1 for simplified example).

Result shows relation between wave traveltime and distance on earth surface. Figure 6.2 shows traveltime for times from 0 to 2500 seconds for all phases. That kind of plot is not very useful due to large amount of phases displayed.

Figure 6.3 shows the same set of information with additional amplitude value, that is presented as color intensity.

For certain purposes code could be modified to record data about particular phases only. That kind of data may by used when searching and identifying phases on seismographs. See example shown on figure 6.4.

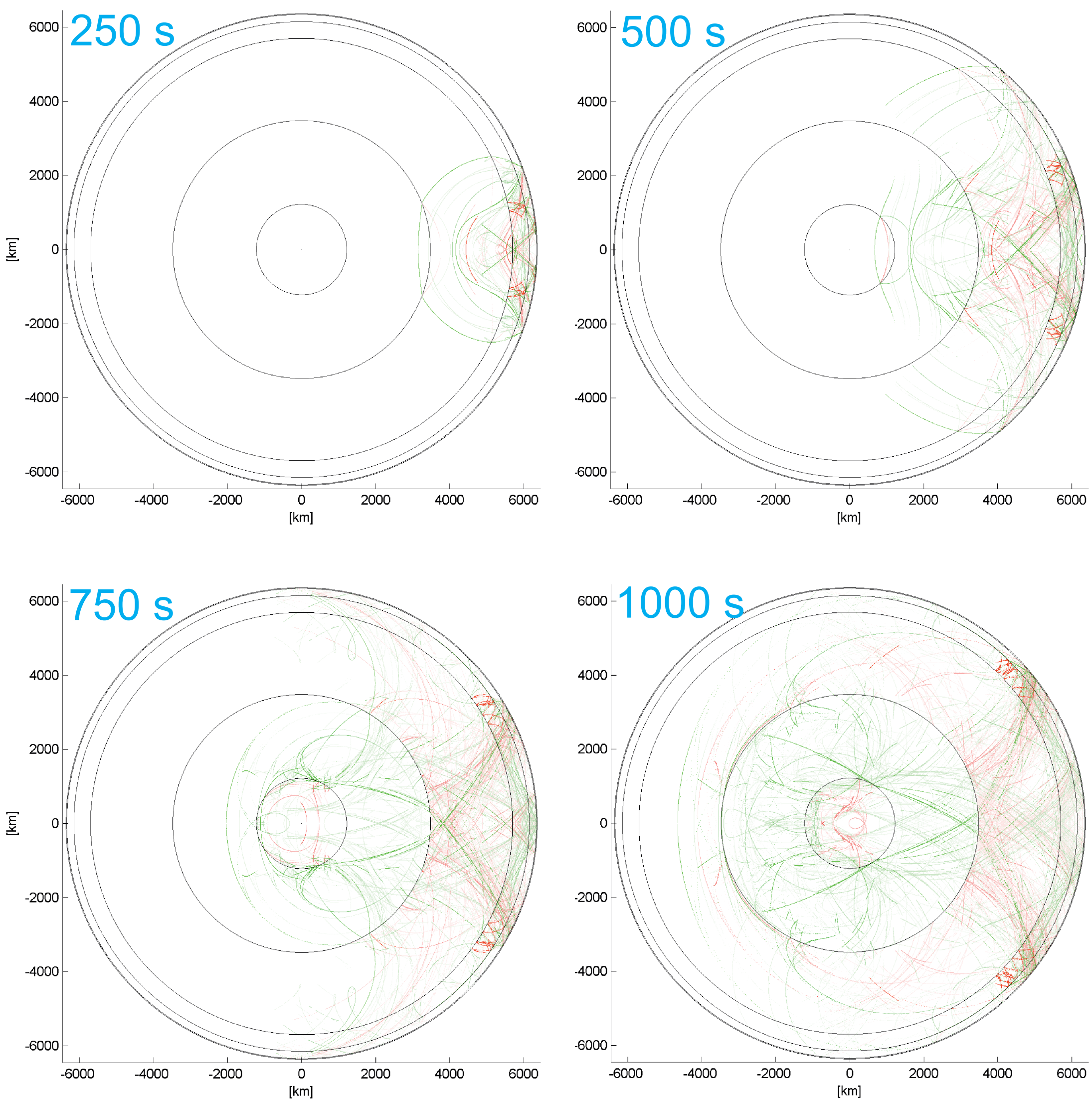

Figure 6.1

Position of wave fronts after 250, 500, 750 and 1000 seconds from event time. Event located at depth of 371 km. Green color corresponds to P-waves while red to S-waves. Note lack of S-waves in area of liquid outer core. Despite that in inner core (solid) S-waves are propagating, because of conversion from P-wave to S-wave on boundary between inner and outer core. Color intensity on plot corresponds to calculated ray amplitude.

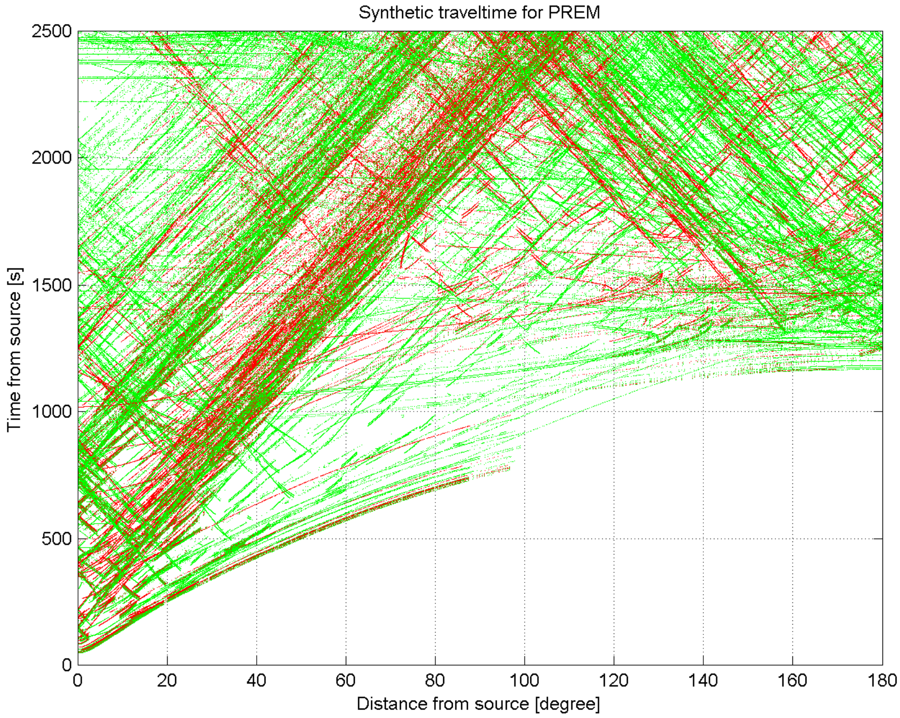

Figure 6.2
Synthetic hodograph from 0 to 2500 seconds without reduction. All phases. Event located at depth of 371 km.

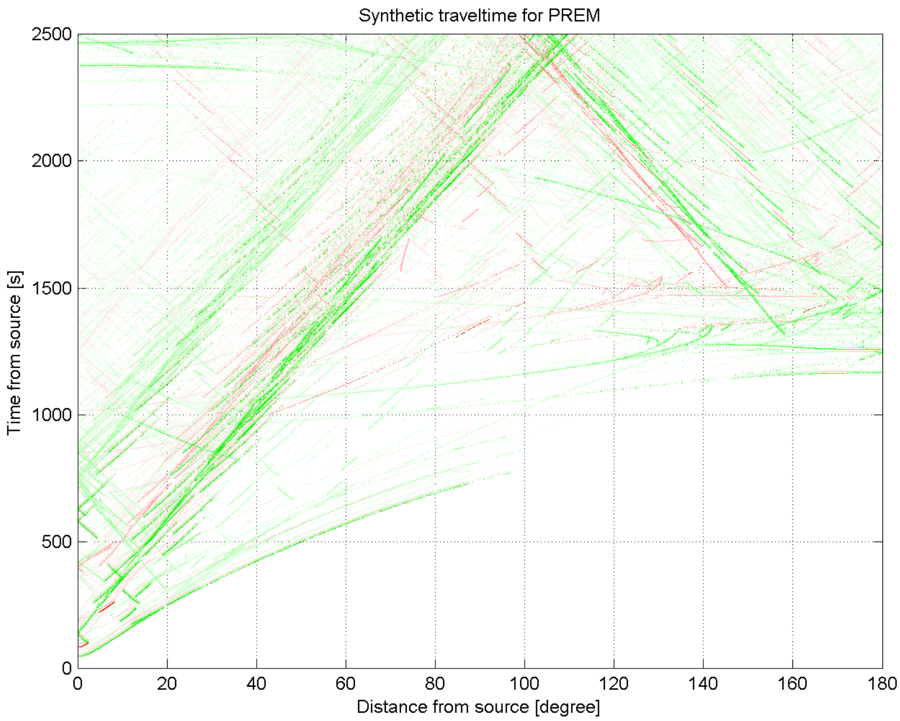

Figure 6.3
Synthetic traveltime from 0 to 2500 seconds without reduction. All phases. Event located at depth of 371 km. Color intensity corresponding to amplitude.

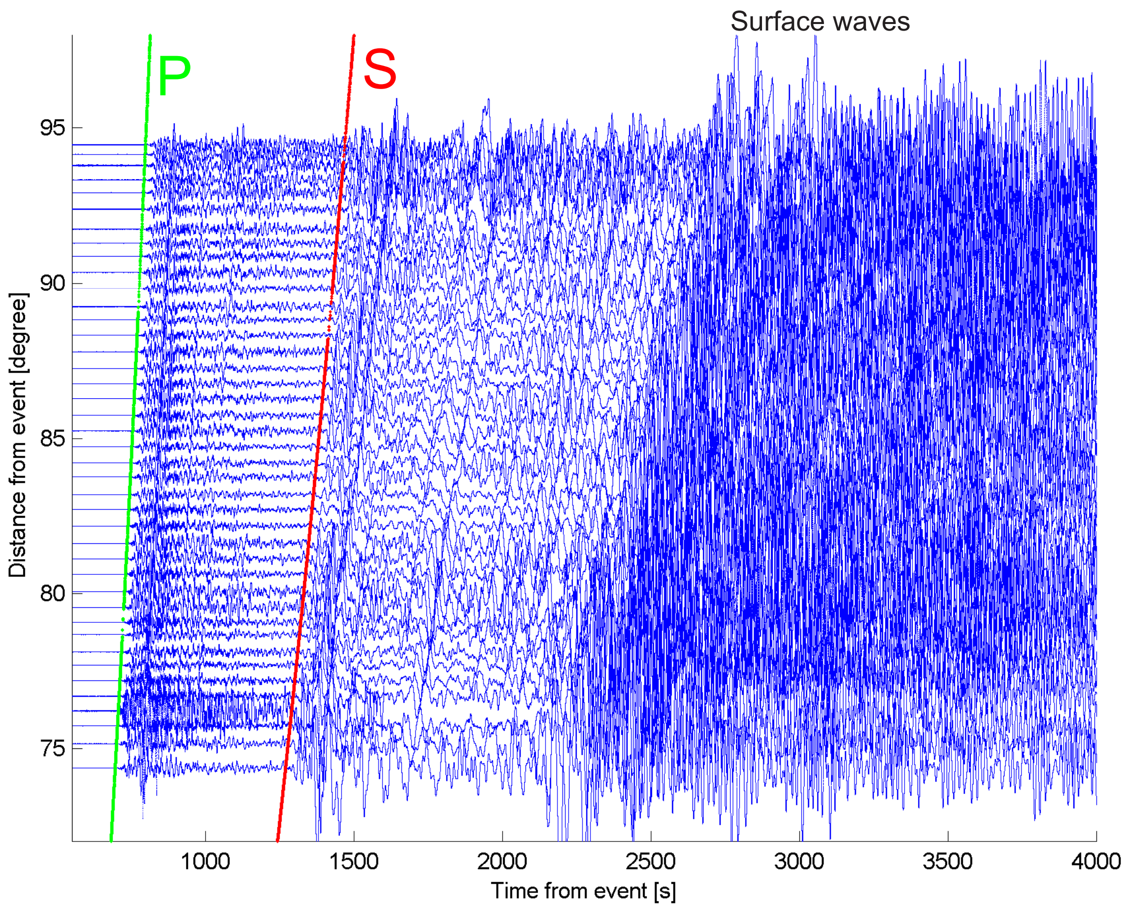

Figure 6.4
Seismic section (only Z-component displayed) recorded by USArray network after earthquake near by Honsiu on 11/03/2011 (magnitude 8.9, located at depth of 24 km). Main P-wave (green) and S-wave (red) phases calculated by program.

6.3 Example 3

Other example of program utilization is map of seismic waves phases in time shown on figure 6.5. This result has, along obvious educational, practical application. It may be used to verify best locations for seismic stations designed to record certain wave phases.

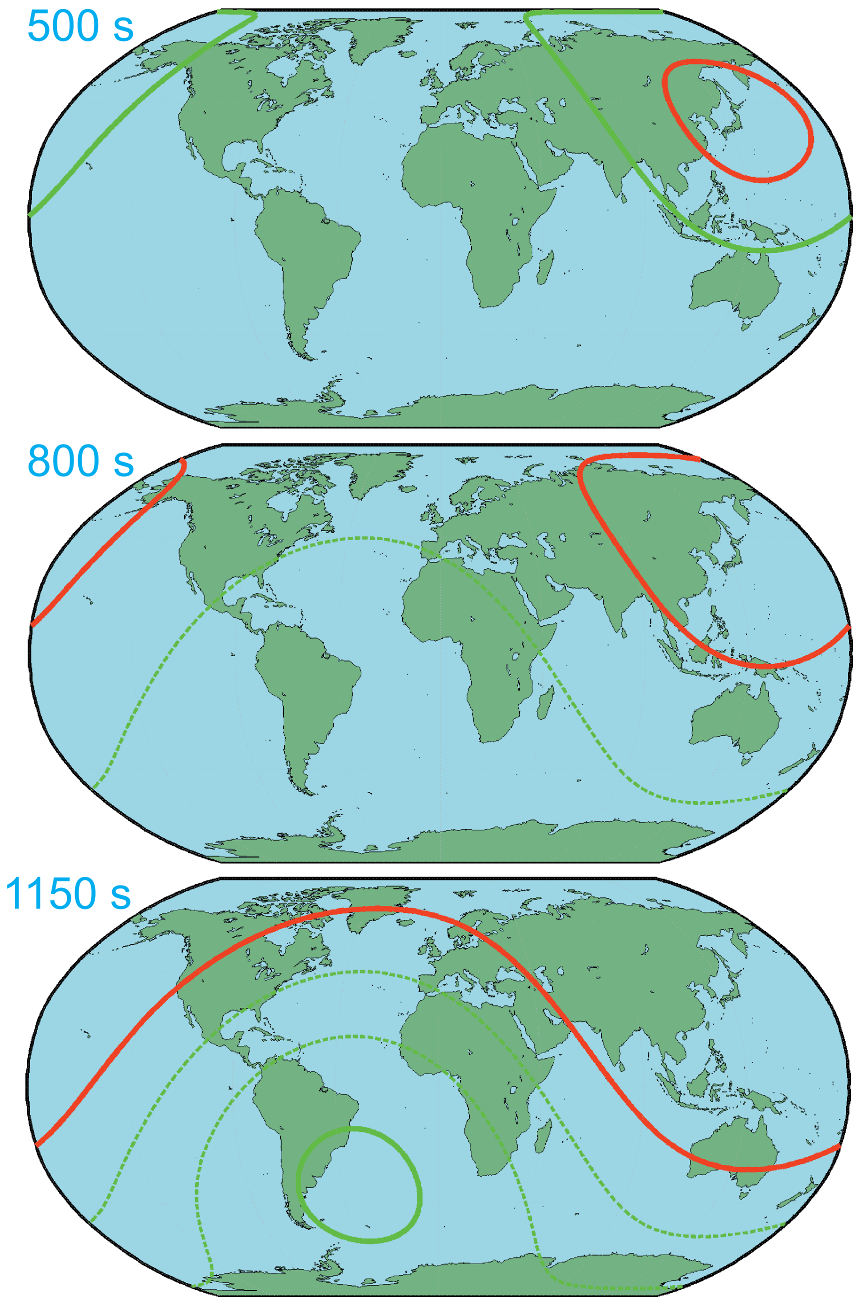

Figure 6.5
Map showing range of P and S-wave after 500, 800 and 1150 seconds from hypothetical earthquake located near by Honsiu at depth of 371 km. Dashed green line on second map indicates range of P-wave propagating through mantle. Note that at this point P-wave front is not visible. Second dashed line on third map indicates beginning of area where P-wave propagating through core can be recorded.

Chapter 7

Conclusions

Simple seismic ray trancing algorithm was developed and successfully implemented. Various extensions and modifications were investigated and found useful. There are several scientific application of such a program, but they can be achieved using by existing programs developed by different universities around the world (note that given results are the same and accurate - difference lays in graphic interface and usability).

The main and most important use of developed program is educational. Generated results have great educational value for high school and university students, and will be placed on Institute of Geophysics web page, so teachers could download and use them.

But watching result itself is only the beginning. Implementing algorithm or modifying existing implementation brings profound understanding of wave propagation process. This allows full comprehension of seismic survey methods: shallow seismic source (there are no earthquakes deeper than 700 km under ground) and registration on surface allow to study whole Earths body because of specific wave route caused by refraction and reflection.

In my opinion shown algorithm implementation should become part of student education for those who are interested in choosing seismology as their career path.

Acknowledgments

I would like to express my gratitude to the following people:

- Prof. Marek Grad for his excellent supervision, patience and support.
- Dr Monika Wilde-Piórko for allowing me to work and expand my knowledge at Lithospheric Physics Department.
- Dr Leszek Czechowski for thorough review.
- Dr Andrzej Wysmołek for great patience presented to all students.
- And anybody I missed who deserves a mention.

Appendix A

Implementation: source code in C++

Compilation of given source code was tested on Windows machine with *g++* (GCC) version 4.5.0 by Free Software Foundation, Inc.

Execution of compiled program does not require any input parameters. After execution program will report time of execution and will save result to file called *result.txt*. Result file should contain 3 numbers in every row separated with tabulator. First value is epicenter distance, second is time and third is type of wave (1: P-wave, 2: S-wave). Beginning of result file shown on listening A.2.

Listing A.1

C++ implementation source code

```cpp
#include <time.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


// Structure of ray input parameters:
struct input_t
{
  float x;
  float y;
  float fi;
  float time_start;
  int WaveType;
};

// Structure of ray output parameters:
struct output_t
{
  int reason;
  float time;
  float dist;
};

input_t *input;
output_t *output;

// Definitions of PREM P and S-wave speed in function of depth:
float PREM_depth[56] = {0, 200, 400, 600, 800, 1000, 1200, 1221.5, 1221.5, 1400, 1600,
    1800, 2000, 2200, 2400, 2600, 2800, 3000, 3200, 3400, 3480, 3480, 3600, 3630, 3630,
    3800, 4000, 4200, 4400, 4600, 4800, 5000, 5200, 5400, 5600, 5600, 5701, 5701, 5771,
    5771, 5871, 5971, 5971, 6061, 6151, 6151, 6221, 6291, 6291, 6346.6, 6346.6, 6356,
    6356, 6368, 6368, 6371};
float PREM_vp[56] = {11266, 11256, 11237, 11206, 11162, 11105, 11036, 11028, 10356,
    10250, 10123, 9985.5, 9835, 9668.6, 9484.1, 9278.8, 9050.1, 8795.7, 8513, 8199.4,
    8064.8, 13717, 13688, 13680, 13680, 13447, 13245, 13016, 12784, 12545, 12293, 12024,
     11734, 11416, 11066, 11066, 10751, 10266, 10158, 10158, 9645.9, 9134, 8905.2,
    8732.1, 8559, 7989.7, 8033.7, 8076.9, 8076.9, 8110.6, 6800, 6800, 5800, 5800, 1450,
    1450};
float PREM_vs[56] = {3667.8, 3663.4, 3650.3, 3628.3, 3597.7, 3558.2, 3510, 3504.3, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7264.7, 7265.8, 7266, 7266, 7188.9, 7099.7,
    7010.5, 6919.6, 6825.1, 6725.5, 6618.9, 6563.7, 6378.1, 6240.5, 6240.5, 5945.1,
    5570.2, 5516, 5516, 5224.3, 4932.6, 4769.9, 4706.9, 4643.9, 4418.9, 4443.6, 4469.5,
    4469.5, 4490.9, 3900, 3900, 3200, 3200, 0, 0};

```

```
// Time step definition
float dt = .01;

// Function returning sign of given number:
float sign(float num)
{
  if(num >= 0)
    return 1.0;
  if(num <= 0)
    return -1.0;
  return 0.0;
}

// Function calculating (interpolating) P or S-wave speed as given depth from PREM model:
float FindSpeed(float depth, int WaveType)
{
  int a;
  if(WaveType == 1) // P-wave
  {
    for(a=0;a<55;a++) // Loop through model given values
    {
      if(depth >= PREM_depth[a] && depth < PREM_depth[a+1]) // if depth between values form model
      {
        return (((depth-PREM_depth[a])*PREM_vp[a+1] + (PREM_depth[a+1]-depth)*PREM_vp[a])/(PREM_depth[a+1]-PREM_depth[a]))/1000.0; // Calculate P-wave speed at given depth
      }
    }
  }
  if(WaveType == 2) // S-wave
  {
    for(a=0;a<55;a++) // Loop through model given values
    {
      if(depth >= PREM_depth[a] && depth < PREM_depth[a+1]) // if depth between values form model
      {
        return (((depth-PREM_depth[a])*PREM_vs[a+1] + (PREM_depth[a+1]-depth)*PREM_vs[a])/(PREM_depth[a+1]-PREM_depth[a]))/1000.0; // Calculate S-wave speed at given depth
      }
    }
  }

  return 0.0; // Depth out of model range or other exception
}

// Function preventing from calculation imaginary trigonometric functions
float ReduceSinf(float value)
{
  if(value > 1)
    return 1.0;

  if(value < -1)
    return -1.0;

  return value;
}

// Function calculating single ray
void CalcRay(input_t *input, output_t *output, int tid)
{
  float x1 = input[tid].x;
  float y1 = input[tid].y;
  int typ = input[tid].WaveType;
  float time = input[tid].time_start;
  float x2 = cos(input[tid].fi);
  float y2 = sin(input[tid].fi);
  float r1 = sqrt(x1*x1+y1*y1);
  float M, sinf, cosf, r2, v2, v1, f1, sinf2, f2, f, theta, ndx, ndy;

  while(r1 < 6367) // Calculate untill surface reached
  {
    // Calculating auxiliary values
    time += dt;
```

```
    M = sqrt(x1*x1+y1*y1) * sqrt(x2*x2+y2*y2);
    sinf = (x1*y2-y1*x2)/M;
    cosf = (y1*y2+x1*x2)/M;
    r1 = sqrt(x1*x1+y1*y1);
    r2 = sqrt((x1+x2)*(x1+x2)+(y1+y2)*(y1+y2));
    v2 = FindSpeed(r2,typ);
    v1 = FindSpeed(r1,typ);

    if(v2 == 0) // Reached area where wave cannot propagate (for example S-wave in liquid medium)
    {
      output[tid].reason = 30; // Set reason of stopping calculation
      output[tid].time = 0;
      output[tid].dist = 0;
      return; // Stop calculation
    }

    // Calculate refraction
    f1 = asin(ReduceSinf(sinf));
    sinf2 = sin(f1) * (v2/v1);
    f2 = asin(ReduceSinf(sinf2));
    f = sign(cosf)*(f2-f1);
    theta = atan2(y2,x2);

    // Calculate new coordinates
    ndx = v1*dt*cos(theta+f);
    ndy = v1*dt*sin(theta+f);

    // Goto new coordinates and calculate next step
    x1 = x1 + ndx;
    y1 = y1 + ndy;
    x2 = ndx;
    y2 = ndy;
  }

  // After reaching surface return time and epicentral distance
  output[tid].reason = 10;
  output[tid].time = time;
  output[tid].dist = atan2(y1,x1);

}

// Main function:
int main(void)
{

  int N = 1024; // Number of rays to calculate

  FILE * pFile; // File structure for data output

  // Execution time is recorded:
  clock_t start, end;
  start = clock();

  // Allocating memory for input and output structs:
  input = (struct input_t*)malloc(N*sizeof(struct input_t));
  output = (struct output_t*)malloc(N*sizeof(struct output_t));

  // Generating start values for all rays:
  for (int i=0; i<N/2; i++) // Fist half of rays
  {
    input[i].fi = (3.14)-i*(2*3.14/(N/2)); // Angular distribution
    input[i].x = 6000; // Starting coordinates
    input[i].y = 0;
    input[i].time_start = 0; // All rays start in event time
    input[i].WaveType = 1; // P-wave
  }
  for (int i=N/2; i<N; i++) // Second half of rays
  {
    input[i].fi = (3.14)-i*(2*3.14/(N/2)); // Angular distribution
    input[i].x = 6000; // Starting coordinates
    input[i].y = 0;
    input[i].time_start = 0; // All rays start in event time
    input[i].WaveType = 2; // S-wave
  }

```

```c
  start = clock(); // Start execution clock

  // Calculate rays (one at the time):
  for (int i=0; i<N; i++)
  {
    CalcRay(input,output, i);
  }

  end = clock(); // Stop execution clock

  printf("czas:\t%f\t", (double)(end-start)/CLOCKS_PER_SEC); // Report execution time

  start = clock(); // Start clock for result saving

  pFile = fopen ("result.txt","w"); // Open result file

  // Save result of every ray to file:
  for (int i=0; i<N; i++)
  {
    // Save only rays that reached surface:
    if(output[i].reason == 10)
    {
      // Save value to file:
      fprintf(pFile,"%f\t%f\t%d\n",output[i].dist,output[i].time,input[i].WaveType);
    }
  }

  fclose (pFile); // Close file

  end = clock(); // Stop clock

  printf( "%f\n", (double)(end-start)/CLOCKS_PER_SEC); // Report saving time

  // Free memory:
  free(input);
  free(output);

  // End
  return 0;
}
```

Listing A.2

Beginning of *result.txt* generated by program

```
3.139909  1165.165527 1
3.100383  1165.005371 1
3.044013  1164.284668 1
3.003553  1163.393799 1
2.955854  1161.952393 1
2.912596  1160.270752 1
2.867654  1158.178711 1
2.819653  1155.526123 1
2.770826  1152.453125 1
2.718101  1148.669434 1
2.663589  1144.295166 1
2.595021  1138.249268 1
2.506549  1129.680908 1
2.498482  1142.273193 1
2.534361  1136.257324 1
2.635098  1148.018799 1
2.613816  1145.276123 1
2.593200  1142.433350 1
2.574962  1139.750732 1
2.557638  1137.108154 1
```

Bibliography

[1] *Poradnik pracownika służby geologicznej.* Wydawnictwo Geologiczne, 1971.

[2] Keiiti Aki and Paul G. Richards. *Quantitative Seismology.* University Science Books, 2002.

[3] Adam M. Dziewonski and Don L. Anderson. Preliminary reference earth model. *Physics of The Earth and Planetary Interiors*, 25(4):297 – 356, 1981.

[4] M. Grad and M. Polkowski. Seismic wave velocities in the sedimentary cover of poland - borehole data compilation. *Acta Geophysica.* Accepted.

[5] A. Guterch, M. Grad, H. Thybo, and G. R. Keller. Polonaise '97 – an international seismic experiment between precambrian and variscan europe in poland. *Tectonophysics*, 314(1-3):101 – 121, 1999.

[6] B. L. N. Kennett and E. R. Engdahl. Traveltimes for global earthquake location and phase identification. *Geophysical Journal International*, 105(2):429–465, 1991.

[7] Dimitri Komatitsch, Gordon Erlebacher, Dominik Göddeke, and David Michéa. High-order finite-element seismic wave propagation modeling with mpi on a large gpu cluster. *Journal of Computational Physics*, 229(20):7692 – 7714, 2010.

[8] Jonathan M. Lees. Zoeppritz equations: R package, 2009.

[9] P. Środa. Seismic anisotropy of the upper crust in southeastern poland - effect of the compressional deformation at the eec margin: Results of celebration 2000 seismic data inversion. *Geophysical Research Letters*, 33, 2006.

[10] V. Červený. *Seismic Ray Theory.* Cambridge University Press, September 2005.